Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118

The RAYMOND simulation package — Generating RAYpresentative MONitoring Data to design advanced process monitoring and control algorithms

Geert Gins[a], Jef Vanlaer[a], Pieter Van den Kerkhof[a], Jan F.M. Van Impe[a,*]

[a] *KU Leuven, Department of Chemical Engineering, Chemical and Biochemical Process Technology and Control (BioTeC), W. de Croylaan 46 PB 2423, B-3001 Leuven, Belgium*

Abstract

This work presents the RAYMOND simulation package for generating RAYpresentative MONitoring Data. RAYMOND is a free MATLAB package and can simulate a wide range of processes; a number of widely-used benchmark processes are available, but user-defined processes can easily be added. Its modular design results in large flexibility with respect to the simulated processes: input fluctuations resulting from upstream variability can be introduced, sensor properties (measurement noise, resolution, range, etc.) can be freely specified, and various (custom) control strategies can be implemented. Furthermore, process variability (biological variability or non-ideal behavior) can be included, as can process-specific disturbances.
In two case studies, the importance of including non-ideal behavior for monitoring and control of batch processes is illustrated. Hence, it should be included in benchmarks to better assess the performance and robustness of advanced process monitoring and control algorithms.

Keywords: process monitoring, process control, data generation, process simulation

1. Introduction

Today's (bio)chemical industry faces a number of important challenges. Globalization and economic crises lead to increased market competition, while climate change results in stricter regulations. Hence, there is a large need for more efficient production plants, producing more goods with less resources and fewer emissions. Advanced Process Monitoring and Control (APMC) provides a solution to this problem.

To accurately assess the performance of APMC algorithms, a large amount of process data is needed during the development, validation and comparison of the techniques. Moreover, a sufficiently rich dataset of *faulty* process data (with knowledge about the exact fault time and preferably also the fault type) is needed when the techniques will be employed for fault detection and diagnosis. Industrial data are not always readily available, especially not when data from faulty operation are required. Deliberately introducing disturbances in the process to create faulty data causes safety issues and has a high cost. Therefore, many researchers make use of data from process simulators. These often use mechanistic models, mostly consisting of differential and algebraic equations, to reproduce the behavior of an industrial process. Variations on the initial conditions, small fluctuations of process inputs and measurement noise are often introduced to simulate run-to-run process variations. Popular tools in literature are the Pensim simulator for penicillin production by filamentous micro-organisms [1] and the simulator for the Tennessee Eastman process [2].

The flexibility of existing simulators, such as the ones mentioned above, is quite limited with respect to the use of different process models, the specification of the amplitude and distribution of measurement noise, the introduction of process faults and the incorporation of control systems. The simulators have been developed to generate data for specific processes with a fixed process layout. To alter the size or type of the measurement noise, one has to dig deep into the simulator code in search for the lines where the noise is generated. The only process faults that can be simulated are the ones that were pre-determined during the simulator development. In order to more correctly benchmark different

*Corresponding author, Tel: +32.16.32.14.66, Fax: +32.16.32.29.91
Email addresses: geert.gins@cit.kuleuven.be (Geert Gins), jan.vanimpe@cit.kuleuven.be (Jan F.M. Van Impe)

techniques for fault detection and rejection, however, a wide range of different process disturbances should be tested.

Moreover, real processes are subject to process variability, which is seldom accounted for in simulators. For example in biochemical processes, small differences in the metabolism of individual micro-organisms or subpopulations of micro-organisms cause (stochastic) variations in the overall growth and production rates. An extreme example of this so-called biological variability is the existence of oscillating yeast cultures [3, 4]. In chemical processes, fouling or impurities may cause variability in heat exchange and other process parameters. The presence of process variability in industrial data may have a severe impact on the monitoring performance of Statistical Process Monitoring (SPM) techniques that are developed and evaluated on the basis of simulation data only.

Therefore, a new simulation package `RAYMOND` was developed, which allows easy introduction of user-specified process models and controllers. Furthermore, full specification of measurement noise and process faults is possible, as is the introduction of process variability as (stochastic) variations of the model parameters.

2. The `RAYMOND` simulation package

This section presents the main functionalities of the new simulation package `RAYMOND`, developed in MATLAB. `RAYMOND` generates RAYpresentative MONitoring Data by enabling the addition of process variability to simulated processes. It is freely available from `http://cit.kuleuven.be/biotec/raymond`.

`RAYMOND` is a modular simulation package. This allows an easy introduction and combination of new process models with any type of controller, process fault, measurement noise and process variability. The flexibility in the choice of controllers and specification of process inputs makes the simulator not only useful for SPM applications, but also for testing of new control algorithms and applications of optimal experimental design.

The main features of `RAYMOND` are

- easy implementation of new processes,
- addition of process variability,
- inclusion of input fluctuations,
- free specification of sensors,
- full control over simulated process faults, and
- easy switching between different control methodologies.

The following sections illustrate these features in more detail.

2.1. Easy implementation of new processes

A wide range of processes can be implemented in `RAYMOND` in a straightforward way. To illustrate this point, two well-known benchmark processes have been implemented: the Pensim simulator, used for SPM [e.g., 5–14], and the Tennessee Eastman process, mainly used for process control and fault detection [e.g., 15–20].

The Pensim model of Birol et al. [1] describes a biochemical fermentation process for the production of penicillin by filamentous micro-organisms on an industrial scale. The reactor initially operates in batch mode, afterwards switching to fed-batch mode to stimulate production of penicillin. The process model contains 9 states and 11 manipulated variables.

The Tennessee Eastman process describes the production of two products from four reactants in an installation consisting of five major unit operations [2]. It is a continuous process described by 30 differential and 160 algebraic states, with various control loops.

Other processes are also available in `RAYMOND`, ranging from single CSTR reactors to distillation columns. Examples include the models from Nihtila and Virkunen [21], Klatt and Engel [22], Henson and Seborg [23], Diehl et al. [24], and Hahn and Edgar [25].

`RAYMOND` can also be used to simulate multi-rate processes (see Section 3.1.2 for more details).

2.2. Addition of process variability

`RAYMOND` supports the inclusion of process variability in the simulation, e.g., to represent oscillations or slow drifts in some model parameters. The effect of adding process variability to the Pensim process—here, it can be interpreted as biological variability—is illustrated in Figure 1. In this example, biological variability is modeled as slow-varying oscillations of the biomass' maximum growth rate.

Figure 1(a) displays the maximum specific growth rate in the base case—without biological variability—and in the case with biological variability, in dashed and full lines, respectively. The corresponding profiles of the penicillin concentration are presented in Figure 1(b).

From these figures, it is observed that biological variability induces more fluctuations in the penicillin concentration. The same effect is observed for other variables, such as dissolved oxygen. Hence, process variability possibly provides an extra challenge for, e.g., detecting process disturbances, predicting or optimizing the batch-end concentration, etc. This will be further illustrated in Section 4.

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118

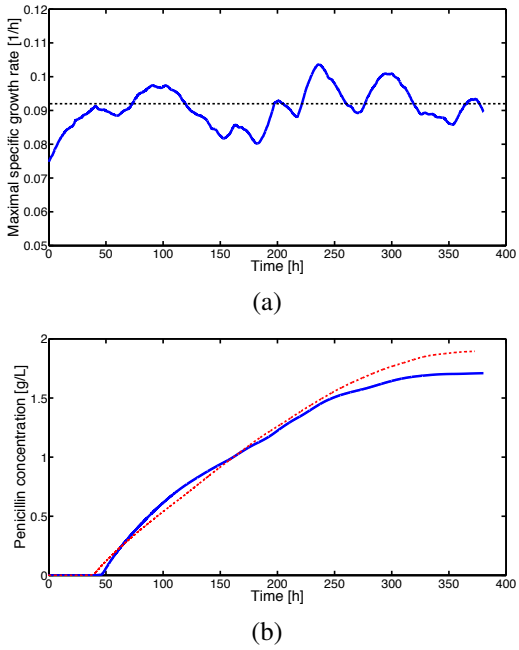

Figure 1: Inclusion and influence of process variability in RAYMOND (- -, without variability; —, with variability). (a) Changing maximum specific growth rate of the biomass. (b) Resulting penicillin concentration. The difference in evolution and extra oscillations prove an extra challenge for process prediction, control, and optimization.

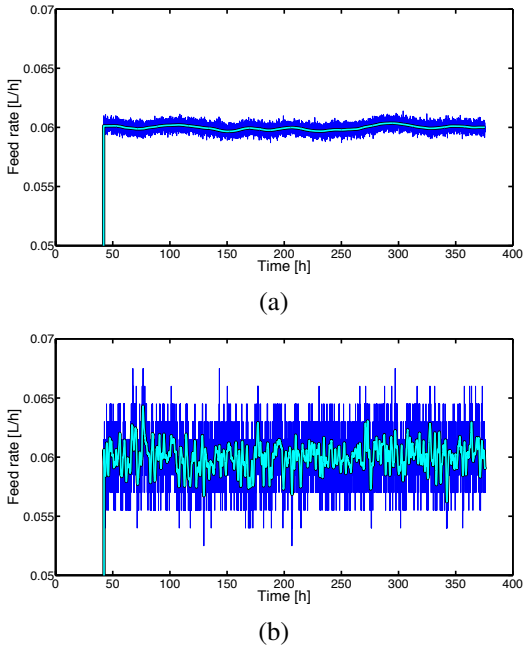

Figure 2: Different types of input variations on the feed rate, as well as different sensor characteristics. The light line depicts the true feed rate; the dark line shows measured values. (a) Slow-varying input variations with Gaussian sensor noise with low amplitude. (b) Fast-varying input variations with Gaussian sensor noise with high amplitude and low sensor resolution.

Changing between different types of process variability only requires changing a few lines of code in the simulation description; changing the underlying process model or actual simulator is not needed. This will be further detailed in Section 3.4.

2.3. *Inclusion of input variations*

The introduction of variations on the input variables of the process is illustrated in Figure 2 for Pensim. Even though no controllers are specified for the feed rate, small oscillations are used to model upstream variability and imperfect equipment behavior. Figure 2(a) depicts the feed rate with small and slow input variations, whereas Figure 2(b) illustrates high-frequency variations of higher amplitude.

Again, switching between the two presented types of input variations is achieved by changing only a few lines in the simulation description.

2.4. *Free specification of sensors*

Figure 2 also illustrates the free specification of sensors with regard to measurement noise and resolution. Figure 2(a) displays Gaussian measurement noise with a small amplitude, whereas Figure 2(b) presents the case with large-amplitude Gaussian noise. The low sensor resolution in the latter case is evident from the quantification of the measured values.

2.5. *Full control over process faults*

The ability to specify various types of process disturbances is presented in Figure 3. The feed rate of the Pensim process is subject to sudden disturbances (step changes) in Figure 3(a). In Figure 3(b), an incipient fault is depicted. This type of disturbance can represent, for example, the failing of a valve: an initial low-amplitude oscillation caused by fouling of the valve followed by oscillations of larger amplitude when part of the valve fails. Finally, Figure 3(c) illustrates that any type of fault can be specified by the user. In this specific case, several different fault types—more specifically, drifts, sudden jumps, slow oscillations, large oscillations, and saturations—are combined to produce a complex profile.

Process faults are easily introduced in the process by selecting them from a library of fault definitions, which can also be expanded with user-defined faults.

Journal homepage: http://www.journals.elsevier.com/computers-and-chemical-engineering/
Original file available at: http://www.sciencedirect.com/science/article/pii/S0098135414002142

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118

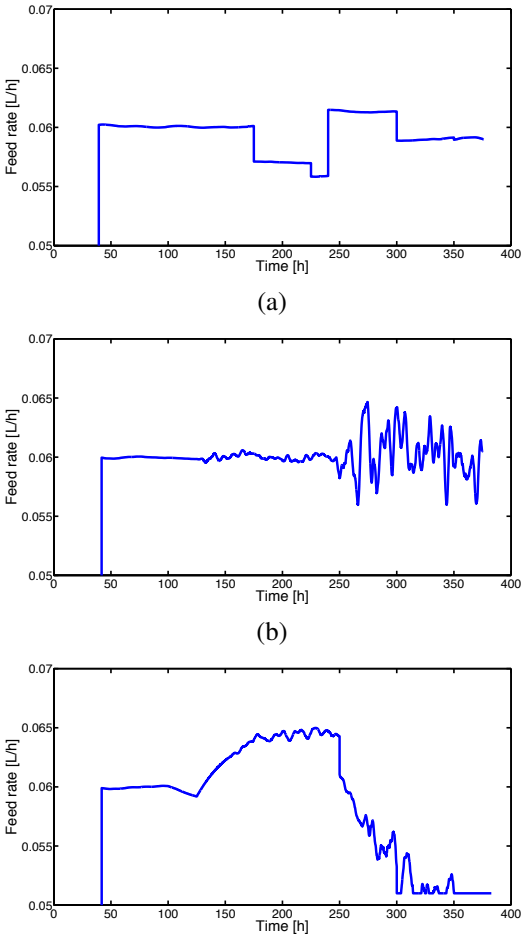

Figure 3: Different types of disturbances can be specified and combined in RAYMOND. (a) Combination of several sudden changes. (b) Incipient fault with small initial amplitude evolving towards oscillations with large amplitude. (c) Several faults types combined.

3. The structure of RAYMOND

Figure 4 presents a simplified graphical representation of the simulation package. The central core of the simulator is formed by the RAYMOND function.

```
data = RAYMOND(ModelName, ...
       SampleInterval, Solver, ...
       InitialConditions, Sensors, ...
       Controllers, SetPoints, ...
       Inputs, ProcessVariability, ...
       Faults, EndCriterion)
```

The 11 inputs of the simulator function form a complete description of the simulation, and have to be provided by the user. The simulation description provides details on the layout of the simulated process, the process conditions, and the properties of the simulation, and is elaborated upon in Section 3.1. Based on this description, the simulator function interacts with several functions in the functions library, which contains model equations, control algorithms, set point definitions, noise distributions, etc. as discussed in Section 3.2. The simulator output is described in Section 3.3. Finally, Section 3.4 details the necessary steps to implement a new process in RAYMOND.

3.1. Simulation description

Before the start of a simulation, the user provides a full description of the simulation in the form of the 11 variables that serve as inputs to the simulator. This section discusses these variables.

3.1.1. Model name

Every simulation is based on a process model, consisting of Ordinary Differential Equations (ODE) and/or Differential Algebraic Equations (DAE) for the state variables. The equations describe the N_S states' mutual interactions and the influence of the N_I model inputs. This process model is contained in the functions library (see Section 3.2).

3.1.2. Sample interval

The sample interval is chosen by the user and is passed to the simulator file. RAYMOND solves the process model between subsequent sample times t_{i-1} and t_i. At the end of each sample interval, new control actions, measurement noise, variability, and fault amplitudes are computed before solving the model for the following interval (from t_i to t_{i+1}).

RAYMOND supports multi-rate processes where the frequencies of different online measurements and control actions are all multiples of a common base rate. In this case, the sample interval should be set equal to the greatest common divisor of the individual sample intervals. For example, a process online temperature measurements every 4 s, pH control actions every 30 s, feed flow rate adjustments every 6 s, and (offline) concentration measurements every few minutes can be simulated with a base sample interval of 2 s.

Because the output of the simulator contains measurements of all state and input variables at the specified (base) rate, the values of less frequent measurements at those times where they are not available in practice should be removed by the user. This also implies that RAYMOND is not suited for processes where sample rates

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118
The content is identical to the published paper, but without the final typesetting by the publisher.

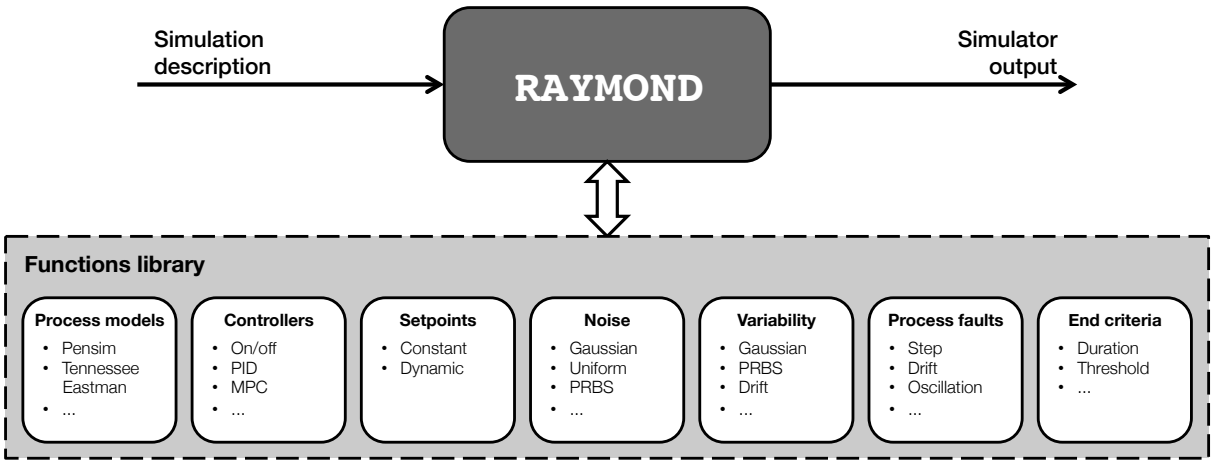

Figure 4: Simplified graphical representation of the RAYMOND simulation package.

differ only slightly, resulting in a base sample interval that is much smaller than the sample intervals of the individual sensors and controllers. An example is a processes where a first variable is sampled every 60 s and a second variable every 61 s, leading to a base sample interval of 1 s. In this case, for every *useful* sample, approximately 59 *useless* samples are generated, each requiring computation power and taking up computer memory.

3.1.3. Solver

The ODE/DAE equations of the process model are solved using any of the available MATLAB solvers. The user specifies the exact solver to be used and its specific options (e.g., a mass matrix to identify differential and algebraic equations in the model).

3.1.4. Initial conditions

The initial conditions for the N_S state variables are specified and passed to the simulator function.

3.1.5. Sensors and measurement noise

RAYMOND offers the possibility to add measurement noise to all measurements of process inputs and state variables. The type and size of the measurement noise is specified for every measurement separately. Again, the noise type corresponds to the name of a function in the functions library (see Section 3.2). Examples include noise with uniform or Gaussian distribution, biased measurements, or autocorrelated measurement noise. If required, the measurement noise can be dependent on one of the process variables. In addition, the range of the sensor output (minimum and maximum values) and its resolution can be defined.

This enables research on the effect of measurement noise and sensor resolution on process monitoring and optimization. For example, it can be tested whether installing more accurate sensors (less noise and/or higher resolution) yields a significant increase in monitoring or optimization performance.

3.1.6. Controllers

A controller is defined for each of the N_I process inputs, describing how the value of each input is determined. The description consists of specifying the controller type, the controller inputs (which states and/or input variables are needed to compute the control action), and controller parameters. The controller type corresponds to a controller function in the functions library (see Section 3.2 for more details), and can range from very simple bang-bang controllers, over PI and PID controllers to more advanced model predictive controllers.

3.1.7. Set points

The information on the set points of all process controllers is passed separately to the simulator function. It defines how the set points corresponding to each controller are computed (corresponding to a set point function in the functions library, cf., Section 3.2). Set point functions can be of any form, from very simple (e.g., constant values) to very complex (e.g., multiple steps at pre-specified times). The parameters and any measurements required for the computation of the set point are also specified by the user.

Journal homepage: http://www.journals.elsevier.com/computers-and-chemical-engineering/
Original file available at: http://www.sciencedirect.com/science/article/pii/S0098135414002142

3.1.8. Inputs and input noise

Upstream variability or non-ideal behavior of process equipment (e.g., small fluctuations of the flow through a pump) are modeled as small variations of the process inputs from their trajectories. Similar to the description of sensors, controllers, and set points, the information on the variations on the N_I inputs is described by the variation type (which again corresponds to a function in the functions library, cf. Section 3.2), the inputs and/or states whose trajectories are needed to compute the fluctuations, and other necessary parameters. The range of each manipulated variable (e.g., due to physical limitations) and its resolution can also be defined.

Input fluctuations can take almost any form. Examples are independent and identically distributed Gaussian fluctuations or slow-varying autocorrelated fluctuations. Changing between different types of input variations makes it possible to investigate the effect of upstream variability on SPM applications or to test the robustness of various monitoring and control strategies.

3.1.9. Process variability

Apart from its flexibility and modularity, RAYMOND's main advantage over other existing simulators is the possibility to include process variability (e.g., non-perfect mixing or biological variability for biochemical processes) in the simulation. Process variability is modelled as variations of the model parameters. The size and type of the variability is specified by the user for each of the N_P parameters susceptible to process variability. Simple examples of process variability are slow oscillations on or drifts of parameters. More complex forms of variability can easily be defined.

The inclusion of process variability is required to correctly benchmark the performance of process monitoring, optimization, and control methodologies for modern production processes. With recent shifts towards biological and biochemical production processes, biological (process) variability will present more challenges for process control, such as the adaptation of organisms to their production environment. Another example is catalytic polymerization, where decreasing catalyst efficiency must be included in the process model to correctly assess the reaction progress.

3.1.10. Process faults

Several types of faults, both static and dynamic, can be introduced in the simulated processes. RAYMOND allows the introduction of sensor faults and faults on the process inputs. Each fault type corresponds to a function in the functions library (see Section 3.2). The user specifies the desired type of fault and required parameters such as the start time, duration, and amplitude. Some examples of typical process faults are sudden step changes, slow drifts, or oscillations, but more complex profiles are easily defined.

Hence, the simulated process upsets can be tailored to the specific application.

3.1.11. End criterion

The stopping criterion defines when the simulated process is terminated. This can, for example, be when data for a pre-determined processing time has been simulated or when one of the state variables surpasses a specific value. Each type of stopping criterion corresponds to a function in the functions library (see Section 3.2), and can be dependent on the system states or input variables.

3.2. Functions library

The functions library comprises the functions that contain process models, provide control algorithms, determine the size of set points, noise and variability or evaluate stopping criteria that are needed for the process simulation. Based on the simulation description by the user, the simulator function calls these functions during the simulation of the envisaged process. The functions library is completely modular: each process model and each type of controller, set point, noise, variability, fault, and stopping criterion is described by a separate function. This way, functions from previous process simulations can readily be re-used and new functions can easily be added to simulate new processes or to include new controller types in an existing process simulation, for example.

3.3. Simulator output

RAYMOND returns a three-dimensional data tensor containing (i) the real values of the process states, (ii) their measured values, and (iii) the specified set points of the process inputs at every sample time. Measurements of inputs or state variables that cannot be measured in industrial practice can be deleted by the user after simulation, but are available in RAYMOND's output to facilitate data analysis.

3.4. Simulating a new process

The flexible structure of the RAYMOND simulation package allows an easy introduction of new processes. To simulate a new process, a new model file has to be added to the functions library, together with functions

for controllers, set points, noise, faults, and stopping criteria that are not yet available.

The following steps should be undertaken:

1. Define the new process model (model equations and parameters) in a model file.
2. Define new (if not yet available) controller types, set point types, noise types, possible process faults, and stopping criteria as separate functions in the functions library.
3. Compose a complete simulation description, as explained in Section 3.1.
4. Simulate the process.

4. Illustration: influence of process variability on Statistical Process Monitoring

This section provides an in-depth case study to demonstrate the impact of process variability on quality prediction and fault detection by means of SPM.

4.1. Description

The impact of process variability on SPM is illustrated using the Pensim [1] benchmark process, implemented in `RAYMOND`. All simulations were conducted in MATLAB 7.11.2 (R2010b SP2) under 64-bit Windows 7 Professional SP1.

Table 1 lists the initial conditions of the various states in the model, as well as the set points for the process inputs. The initial substrate concentration, biomass concentration, and culture volume are subject to random variations for each batch to represent changing initial conditions; they are sampled from a normal distribution with the 95% confidence intervals indicated in Table 1. Additionally, small low-frequency fluctuations are added to several process inputs to represent a real process environment. Reactor temperature and pH are controlled at their respective set points by standard PID controllers during both phases. For the full model equations, parameter values and controller tunings, the reader is referred to the original Pensim paper of [1]. Table 2 provides an overview of the available on- and offline measurements. For each batch, the data acquired by the 11 online sensors is aligned to a length of 602 samples using the indicator variable technique, following the procedure of [26]. Next, time is added as a 12th variable to retain speed information, resulting in a total of 7224 online measurements for each batch. The fermentation is stopped when a total of 25 L of substrate feed has been added to the reactor. The concentration of penicillin at batch completion is the parameter on which the quality of the batch is assessed.

Table 1: Initial conditions of the state variables and set points of the process inputs in Pensim.

Initial conditions	Value
Substrate concentration [g/L]	17.5 ± 2
Biomass concentration [g/L]	0.125 ± 0.06
Volume [L]	102.5 ± 10
Dissolved oxygen concentration [g/L]	1.16
Penicillin concentration [g/L]	0
CO_2 concentration [g/L]	0.4487
pH	5
Reactor temperature [K]	298
Reaction heat [cal]	0
Process inputs	Set point
Substrate feed rate [L/h]	0.06
Aeration rate [L/h]	8
Agitator power [W]	30
Feed temperature [K]	296
Controlled variables	Set point
Reactor temperature [K]	298
pH	5

4.2. Process variability in the Pensim model

In the original Pensim equation, the biomass behavior is purely deterministic. In practice however, biological variability results in some stochastic behavior of the organisms. Therefore, process (biological) variability is introduced in the Pensim process as a variation of the biomass's maximum specific growth rate μ_X. The specific growth rate μ, which determines the increase of the biomass concentration, depends linearly on μ_X, and is, therefore, directly impacted by the biological variability. Substrate and dissolved oxygen (DO) concentration are also dependent on μ. The biomass growth also influences the CO_2 concentration and pH. Via the reaction heat associated with growth, the reactor temperature is influenced, so that the biological variability—via interdependencies in the Pensim differential equations—influences the trajectories of almost all state variables.

In this case study, two different models are used for biological variability.[1] The first approach models biological variability as a low-frequency auto-correlated

[1] The authors do not claim either model is correct for modeling biological variability in the Pensim process. The different models are only used for illustrative purposes.

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118
The content is identical to the published paper, but without the final typesetting by the publisher.

Table 2: Measurements available from Pensim.

Online measurements	
Dissolved oxygen conc.	Agitator power
Volume	Feed temperature
pH	Water flow rate
Reactor temperature	Base flow rate
Substrate feed rate	Acid flow rate
Aeration rate	
Offline (quality) measurement	
Penicillin concentration	

variation of μ_X around its value of 0.092 h^{-1}. The size of the biological variability is specified by the standard deviation σ_{bv}. It is obtained by averaging a PRBS signal that randomly alternates between $\sqrt{1000} \times \sigma_{bv}$ and $-\sqrt{1000} \times \sigma_{bv}$ over 1000 samples using MATLAB's `filtfilt` function. This type of variability can be caused by, for example, non-perfect mixing of the reactor medium.

In the second approach, biological variability is modeled as a linear increase of the maximum specific growth rate, starting from its nominal value of 0.092 h^{-1}. The size of the variability is characterized by the increase δ_{bv} of the growth rate over 500 h of operation. This type of variability might represent gradual adaptation of the micro-organisms to their environment.

4.3. Batch-end quality prediction

The first case study explores how biological variability impacts the accuracy of multivariate batch-end quality prediction via Partial Least Squares (PLS)[27].

In this study, 11 different levels of biological variability are considered for the slow variations of the growth rate: $\sigma_{bv} \in \{0, 0.001, 0.002, \ldots, 0.010\}$. A total of 200 batches under Normal Operating Conditions (NOC) are generated at each level. Subsequently, a PLS model is constructed to infer the final penicillin concentration from the online measurements, and the evolution of both the offline and online quality predictions are investigated. Hereto, four-fold crossvalidation is conducted to obtain PLS predictions of the final penicillin concentration for all 200 available batches. In each fold 150 batches are used for training, and 50 for validation. The crossvalidation Sum of Squared Errors (SSE) is used as a measure of predictive quality.

The same procedure is followed for the biological variability represented by a linearly increasing growth rate. Next to the case without variability ($\delta_{bv} = 0$), four levels of variability are considered, which cause the biomass' growth rate to increase with, respectively, δ_{bv} = 1–2%, 2–4%, 5–10%, and 10–20% over 500 h. Within each range, δ_{bv} follows a uniform distribution.

4.3.1. Mathematical tools

A Partial Least Squares (PLS) [27] model relates the online measurements of each batch to the corresponding quality. Batch-wise unfolding [28] obtains a 150×7224 training data matrix $\mathbf{X}_{tr}$, where row i of $\mathbf{X}_{tr}$ contains all measurements at all time points of batch i. The columns of $\mathbf{X}_{tr}$ are mean-centered and scaled to unit variance to remove most nonlinearities from the data. The quality matrix $\mathbf{Y}_{tr}$ of size 150×1 is also mean-centered and scaled to unit variance.

PLS projects the in- and output matrices onto a space of low dimension $R \ll 7224$, characterizing each batch with R instead of 7224 variables. Next, PLS finds a linear relation between both.

$$\begin{cases} \mathbf{X}_{tr} = \mathbf{T}_{tr}\mathbf{P}^\top + \mathbf{E}_\mathbf{X} \\ \mathbf{Y}_{tr} = \mathbf{T}_{tr}\mathbf{Q}^\top + \mathbf{E}_\mathbf{Y} \end{cases} \tag{1}$$

The projections are defined by the model loading matrices $\mathbf{P}$ ($7224 \times R$) and $\mathbf{Q}$ ($1 \times R$) for in- and output space respectively. The scores matrix $\mathbf{T}$ ($150 \times R$) is the representation of the 150 training batches in the reduced space for each of the crossvalidation folds. The matrices $\mathbf{E}$ contain the modeling residuals. The loadings $\mathbf{P}$ and $\mathbf{Q}$ are obtained via the NIPALS algorithm [27], and are chosen in such a way that as much covariance between $\mathbf{X}_{tr}$ and $\mathbf{Y}_{tr}$ as possible is retained.

During model identification, a weight matrix $\mathbf{W}$ ($7224 \times R$) with orthonormal columns is determined to compute a (new) batch's scores $\mathbf{T}$ ($1 \times R$) given a measurement set $\mathbf{X}$ (1×7224).

$$\mathbf{T} = \mathbf{X}\mathbf{W}\left(\mathbf{P}^\top\mathbf{W}\right)^{-1} \triangleq \mathbf{X}\mathbf{B} \tag{2}$$

Combining (1) and (2) leads to the relation between the estimated output $\hat{\mathbf{Y}}$ and the input $\mathbf{X}$.

$$\hat{\mathbf{Y}} = \mathbf{T}\mathbf{Q}^\top = \mathbf{X}\mathbf{B}\mathbf{Q}^\top \tag{3}$$

When monitoring a new batch online, however, $\mathbf{X}$ is only partially known at time t because the future measurements are evidently unknown. Trimmed Score Regression (TSR) [12, 29] is used to compensate for the missing future measurements, yielding the following expression for estimating $\mathbf{Y}$ online.

$$\hat{\mathbf{Y}}(t) = \mathbf{X}_t\mathbf{B}_t\left(\mathbf{B}_t^\top\mathbf{X}_{tr,t}^\top\mathbf{X}_{tr,t}\mathbf{B}_t\right)^{-1}\mathbf{B}_t^\top\mathbf{X}_{tr,t}^\top\mathbf{X}_{tr}\mathbf{B}\mathbf{Q}^\top \tag{4}$$

Journal homepage: http://www.journals.elsevier.com/computers-and-chemical-engineering/
Original file available at: http://www.sciencedirect.com/science/article/pii/S0098135414002142

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118

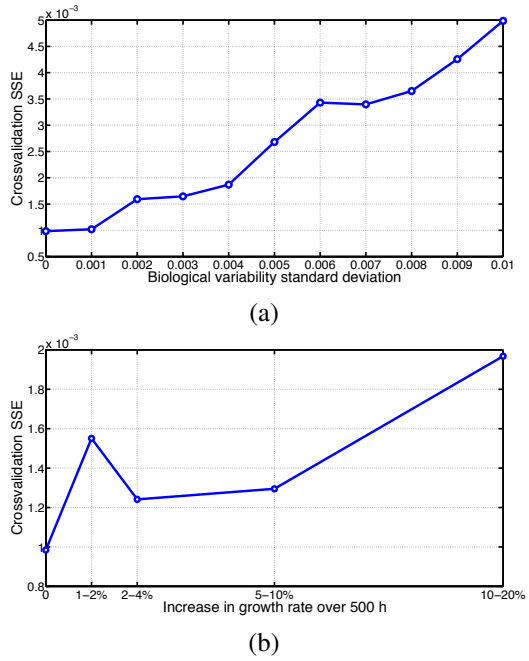

Figure 5: Offline four-fold crossvalidation SSE for prediction of batch-end penicillin concentration over 200 normal batches as a function of the size of the biological variability for (a) the slow variations of the growth rate and (b) the linearly increasing growth rate.

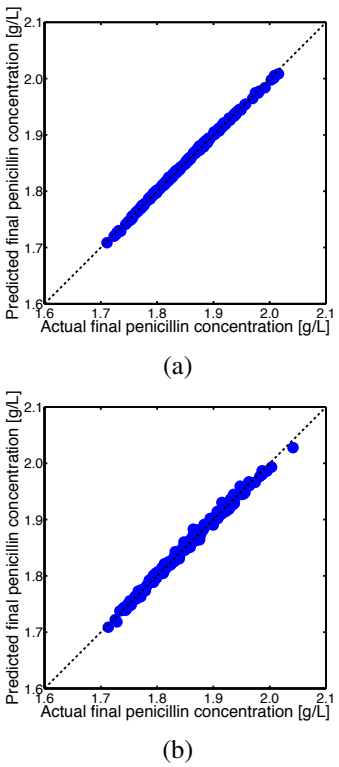

Figure 6: Predicted versus actual final penicillin concentration over 200 batches in four-fold crossvalidation for data containing (a) no biological variability and (b) slow variations of the growth rate with a standard deviation $\sigma_{\text{bv}} = 0.01$.

$\mathbf{X}_t$, $\mathbf{X}_{\text{tr},t}$, and $\mathbf{B}_t$ contain, respectively, the columns of $\mathbf{X}$, the columns of $\mathbf{X}_{\text{tr}}$, and the rows of $\mathbf{B}$ that correspond with the time points 1 through t (in this case, the first $12t$ columns/rows).

The number of latent variables R of the PLS models is selected using the adjusted Wold's R criterion with a threshold $\alpha = 0.95$ as the R for which the following equation holds [30].

$$\frac{\text{SSE}_{\text{cv}}(R+1)}{\text{SSE}_{\text{cv}}(R)} > 0.95 \qquad (5)$$

with $\text{SSE}_{\text{cv}}(R)$ the crossvalidation SSE of the PLS model with R components over the 150 training batches.

4.3.2. Results

Offline and online batch-end quality prediction are presented separately, followed by a short general discussion of the results.

Offline prediction. In offline prediction, the penicillin concentration is estimated after batch completion, when all measurements are known. Hence, no compensation for missing measurements is needed.

For the slow variations of the growth rate, a PLS model with 6 latent variables—as determined using Equation (5)—is constructed for each of the 11 levels of the biological variability. Figure 5(a) plots the (four-fold) crossvalidation SSE as a function of the standard deviation of the biological variability. As the level of biological variability increases, the prediction performance of the MPLS models is slightly worsened. Without biological variability, an SSE of 9.8×10^{-4} is obtained while a biological variability standard deviation σ_{bv} of 0.01 leads to an SSE of 5.0×10^{-3}. However, good predictions are obtained in both cases, as evidenced by Figure 6, which compares the predicted batch-end penicillin concentrations (in four-fold crossvalidation) to the actual values for all 200 available batches for the case without biological variability and for biological variability with a standard deviation σ_{bv} of 0.01 (the highest level tested). The spread of the plotted points around the bisector is slightly higher for the case with biological variability, which suggests that the presence of biological variability slightly deteriorates the predictions. However, the accuracy of the predictions is still more than acceptable.

Journal homepage: http://www.journals.elsevier.com/computers-and-chemical-engineering/
Original file available at: http://www.sciencedirect.com/science/article/pii/S0098135414002142

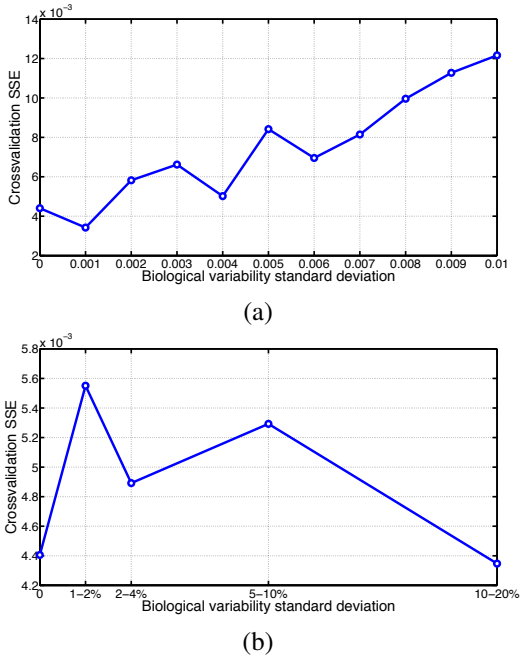

Figure 7: Four-fold crossvalidation SSE for prediction of batch-end penicillin concentration over 200 normal batches after $t = 200$ samples, using TSR for compensation of missing future measurements, as a function of the size of the biological variability for (a) the slow variations of the growth rate and (b) the linearly increasing growth rate.

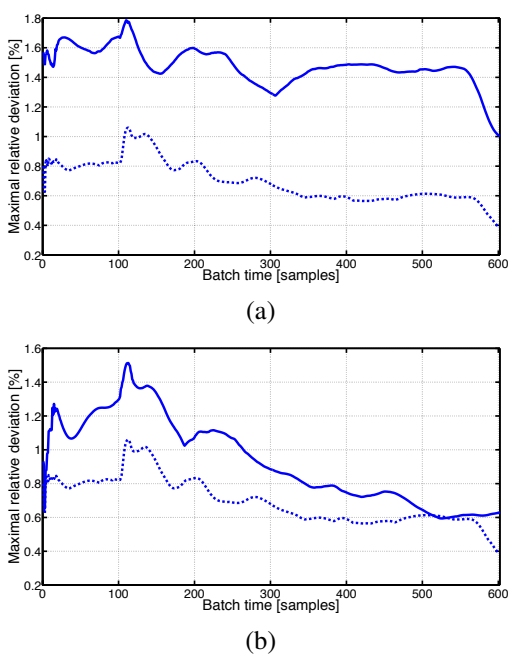

Figure 8: Maximal relative deviation of the batch-end quality prediction from its real value over 200 NOC batches in four-fold crossvalidation as a function of the batch time. (a) Slow variations of the growth rate without biological variability $\sigma_{\text{bv}} = 0$ (- -) and with $\sigma_{\text{bv}} = 0.01$ (—). (b) Linearly increasing growth rate without biological variability $\delta_{\text{bv}} = 0$ (- -) and with $\delta_{\text{bv}} = 10 - 20\%/500h$ (—).

For the linearly increasing growth rate, Eq. (5) also indicates using 6 latent variables for each of the investigated levels of biological variability. The evolution of the crossvalidation SSE as a function of the variability is presented in Figure 5(b). This graph leads to similar conclusions as Figure 5(a): increasing biological variability leads to poorer prediction accuracy. For the Pensim process, however, the accuracy of the model estimates is still acceptable even for the highest tested variability levels.

Online prediction. Using TSR to compensate for missing future measurements, online predictions of the final penicillin concentration are made while the batch is still running.

Figure 7 depicts the crossvalidation SSE as a function of the size of the biological variability after 200 samples for both types of variability. For the slow variations in growth rate, Figure 7(a) is very similar to Figure 5(a): a slightly increasing trend is observed. For the variations in growth rate with a standard deviation σ_{bv} of 0.01, a crossvalidation SSE of 1.2×10^{-2} is obtained, three times higher than the SSE of 4.4×10^{-3} in the case without biological variability. For the linearly increasing growth rate in Figure 7(b), no clear trend is observed.

The online SSE values (Figure 7) are evidently higher than the SSE values for offline prediction (Figure 5) since the TSR compensation for future measurements leads to higher uncertainty. However, the obtained predictions are still very good at all levels and for both types of the biological variability.

To further assess the accuracy of the online prediction, the relative deviation of the online estimate from the actual final penicillin concentration of the batch is investigated. Figure 8 presents the maximal relative deviation—a measure for the worst-case performance—over the 200 available batches (in crossvalidation) as a function of time for the case without biological variability and for the highest variability level tested ($\sigma_{\text{bv}} = 0.01$ in Figure 8(a) and $\delta_{\text{bv}} = 10 - 20\%$ in Figure 8(b)). The maximal relative deviation decreases with time as less compensation for missing future measurements is needed. When biological variability is present, the predictive performance is consistently deteriorated. However, even in the worst-case scenario, the predictions de-

Postprint version of paper published in Comput. Chem. Eng. 2014, 69:110–118
The content is identical to the published paper, but without the final typesetting by the publisher.

viate less than 2% from the actual final quality throughout the complete batch for both types of variability.

4.4. Fault detection

The second case study illustrates the impact of process variability on multivariate fault detection via Principal Component Analysis (PCA) [31]-based techniques. The evolution of the detection delay and false alarm rates is investigated.

Four levels are considered for each type of variability: standard deviations σ_{bv} of 0 (no variability), 0.001, 0.005, and 0.010 for the slow variations in growth rate, and increases δ_{bv} of 0% (no variability), 1–2%, 5–10%, or 10–20% over 500 h for the linearly increasing growth rate; δ_{bv} follows a uniform distribution within each range.

For each type and at each level of variability, 150 NOC training batches are simulated. Next, a PCA model is identified from the NOC batches and subsequently used to monitor the process online and detect process faults. Two kinds of process faults are considered: (i) a gradual decrease of the feed rate with 0.05%/h, starting after 100 h and (ii) a negative drift of the DO sensor with 0.05%/h, also starting after 100 h. Per fault, 50 batches are generated for each type and level of variability.

4.4.1. Mathematical methods

Fault detection consists of two steps. First, NOC process behavior is characterized. Next, fault statistics are used to detect changes from NOC.

In this study, PCA [31] is used to model the process under NOC. As in Section 4.3, batch-wise unfolding is employed to yield a 150×7224 training data matrix $\mathbf{X}_{\text{tr}}$, which is then mean-centered and scaled to unit variance. Similar to PLS, PCA projects $\mathbf{X}_{\text{tr}}$ to a space of dimension $R \ll 7224$. However, PCA only considers the input space $\mathbf{X}_{\text{tr}}$.

$$\mathbf{X}_{\text{tr}} = \mathbf{T}_{\text{tr}}\mathbf{P}^{\top} + \mathbf{E}_{\mathbf{X}} \tag{6}$$

The resulting scores matrix $\mathbf{T}_{\text{tr}}$ ($150 \times R$) is the low-dimensional approximation of $\mathbf{X}_{\text{tr}}$. The loadings matrix $\mathbf{P}$ ($7224 \times R$) defines the projection from the original to the reduced measurement space; its columns are the eigenvectors corresponding to the R largest eigenvalues λ_r of the 7224×7224 covariance matrix of $\mathbf{X}_{\text{tr}}$. Modeling errors are contained in the residual matrix $\mathbf{E}_{\mathbf{X}}$ (150×7224).

Because the columns of $\mathbf{P}$ are orthonormal, the scores $\mathbf{T}$ ($1 \times R$) of a (new) batch can be readily computed from its measurement matrix $\mathbf{X}$ (1×7224).

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{7}$$

For online estimation of a batch's scores, TSR is again employed.

$$\hat{\mathbf{T}}(t) = \mathbf{X}_t\mathbf{P}_t\left(\mathbf{P}_t^{\top}\mathbf{X}_{\text{tr},t}^{\top}\mathbf{X}_{\text{tr},t}\mathbf{P}_t\right)^{-1}\mathbf{P}_t^{\top}\mathbf{X}_{\text{tr},t}^{\top}\mathbf{X}_{\text{tr}}\mathbf{P} \tag{8}$$

$\mathbf{P}_t$ contains the rows of $\mathbf{P}$ corresponding to the first t time points.

Once the scores $\hat{\mathbf{T}}$ of the batch are known, the two fault detection statistics are computed. Hotelling's T^2 is a measure of the distance from the new batch's scores $\mathbf{T}$ to region of the NOC scores. The *SPE* (Squared Prediction Error) monitors the residuals of the new batch and measures the distance of the new batch $\mathbf{X}$ to the PCA hyperplane.

$$T^2(t) = \hat{\mathbf{T}}(t)\,\Sigma_T(t)^{-1}\,\hat{\mathbf{T}}(t)^{\top} \tag{9}$$

$$SPE(t) = \left(\mathbf{X}_\theta - \hat{\mathbf{T}}(t)\mathbf{P}_\theta^{\top}\right)\left(\mathbf{X}_\theta - \hat{\mathbf{T}}(t)\mathbf{P}_\theta^{\top}\right)^{\top} \tag{10}$$

$\Sigma_T(t)$ ($R \times R$) is the covariance matrix of the NOC scores estimated at time t via crossvalidation. $\mathbf{X}_\theta$ and $\mathbf{P}_\theta$ respectively contain the columns of $\mathbf{X}$ and rows of $\mathbf{P}$ that correspond with the current time point t.

The 150 training batches are used to compute an upper control limit with tolerance level α (e.g., $\alpha = 99.9\%$) for both statistics [32]. When either of the statistics exceeds its corresponding control limit u, a disturbance in the process is detected.

$$u_{T^2} = \frac{R\left(150^2 - 1\right)}{150\,(150 - R)}\,F_{(R,150-R;\alpha)} \tag{11}$$

$$u_{SPE}(t) = \frac{\sigma^2(t)}{2\,\mu(t)}\,\chi^2_{\left(2\mu^2(t)/\sigma^2(t);\alpha\right)} \tag{12}$$

$F_{(R,150-R;\alpha)}$ is the upper critical value of the F-distribution with R numerator and $150 - R$ denominator degrees of freedom and a tolerance level α. $\mu(t)$ and $\sigma^2(t)$ are the mean and variance of the *SPE* for the (cross-validated) training set at time t, and correspond with normal, in-control operation. The upper critical value of the χ^2-distribution with $2\mu^2/\sigma^2$ degrees of freedom and tolerance level α is denoted $\chi^2_{\left(2\mu^2/\sigma^2;\alpha\right)}$. u_{T^2} is constant in time, while u_{SPE} is time-varying.

The number of principal components R in this case study is determined using an adaptation of the adjusted Wold's R criterion [30], choosing the smallest R for which the following equation holds (after ranking the eigenvalues λ_r and corresponding eigenvectors from largest to smallest).

Journal homepage: http://www.journals.elsevier.com/computers-and-chemical-engineering/
Original file available at: http://www.sciencedirect.com/science/article/pii/S0098135414002142

$$\frac{\sum_{r=1}^{R+1} \lambda_r}{\sum_{r=1}^{R} \lambda_r} < 1.05 \tag{13}$$

4.4.2. Results

For the variability modeled as slow variations in growth rate, Equation (13) yields 4 principal components at all four levels of the biological variability. Table 3 summarizes the fault detection results at a significance level α of 99.9% for both the gradual feed rate decrease and the gradual DO sensor failure. The table gives an overview of the mean detection delays over the 50 validation batches at the four biological variability levels, the standard deviation on the detection delays and the p-values of standardized t-tests to compare the mean detection time at all levels. In all tested cases, the *SPE* detects the disturbance faster than T^2. Although the gradual DO sensor failure is detected faster and with less spread on the detection times, the general observations with respect to the influence of biological variability are similar for both faults. From Table 3, it is clear that the presence of biological variability in the data leads to a significantly faster fault detection. This is corroborated by the very low p-values for comparison of the mean detection time with the case without biological variability (p_0), which provide sound statistical evidence to conclude that the average fault detection time changes when biological variability is included in the process data. For the gradual DO sensor failure, it is clear that the mean detection time decreases with increasing biological variability. For the feed rate decrease, this evolution is less pronounced but still present.

The fault detection results for the linearly increasing growth rate are reported in Table 4. Again, the presence of biological variability leads to faster fault detection. Contrary to the slow variations of the growth rate, however, no further decrease of the detection delay is observed for increasing levels of variability.

The number of false alarms is another important performance indicator in fault detection. False alarms are only obtained during one batch at the highest level of biological variability σ_{bv} for the DO sensor failure when variability is modeled as slow variations in growth rate. For the linearly increasing growth rate, one batch suffers from a short false alarm at each level of variability $\sigma_{\text{bv}} \in \{1-2\%, 5-10\%, 10-20\%\}$. These results are in line with the 99.9% control limit. Hence, no influence of the biological variability on the false alarm rate is observed.

4.5. Discussion

Two case studies were conducted to demonstrate the importance of process variability for SPM benchmarks.

Predictions of batch-end quality (both online and offline) were worsened when biological variability is present in the process, both for offline and online prediction studies. The relative difference in prediction performance between data with and without process variability remains more or less constant with time. Although the effect was small and satisfactory predictions were still obtained at all levels of the biological variability in the presented case study, this result cannot be generalized. Process variability may have a larger influence on batch-end quality prediction in other processes. Nonetheless, a negative impact of process variability for batch-end quality prediction was clearly demonstrated. This leads to poorer estimation of future process behavior and product quality when testing quality prediction (e.g. batch-end quality prediction), process optimization (e.g., optimization recipe tracking), and process control (e.g., MPC control of final product quality).

Detection of process disturbances was improved when biological variability is present in the data. For the slow variations of the growth rate, the detection delay of both investigated fault types decreased for increasing levels of variability. When biological variability was modeled as an increasing growth rate, any amount of variability was sufficient to improve detection. For both cases, a positive effect of process variability on fault detection was observed.

It is concluded that variability has an important but application-specific effect, sometimes positive and sometimes negative. Hence, it is concluded that process variability should be included in the data when benchmarking APMC to obtain realistic performance estimates and fair comparisons. This is especially true when testing SPM techniques that combine fault detection and quality prediction, such as the PLS-based fault detection and quality prediction originally proposed by Nomikos and MacGregor [33] and its derivatives—recent overviews are presented by Qin [34] and Ge et al. [35].

A second, counterintuitive observation is that the presence of process variability resulted in a better fault detection where a performance degradation would be expected instead based on the reasoning that variability obfuscate process dynamics, making process monitoring more difficult. This result should be the subject of further investigation. It is hypothesized that the autocorrelated nature of the process variability in the case study leads to persistent excitation of the process, caus-

Table 3: Fault detection results for the slow variations in growth rate: mean and standard deviation of the detection delay, and p-values of the standardized t-tests that compare the mean detection time to the case without biological variability (p_0), with variability with $\sigma_{bv} = 0.001$ ($p_{0.001}$), and with variability with $\sigma_{bv} = 0.005\%$ ($p_{0.005}$).

σ_{bv}	Detection [h]		p-value		
	μ	σ	p_0	$p_{0.001}$	$p_{0.005}$
Gradual feed rate decrease					
0	31.1	5.6	–	–	–
0.001	26.8	5.4	2.0×10^{-4}	–	–
0.005	25.3	3.8	3.8×10^{-8}	1.2×10^{-1}	–
0.01	23.2	5.7	3.6×10^{-10}	1.5×10^{-3}	2.9×10^{-2}
Gradual DO sensor failure					
0	21.4	0.8	–	–	–
0.001	19.9	1.1	3.1×10^{-12}	–	–
0.005	17.8	1.3	1.9×10^{-30}	3.6×10^{-14}	–
0.01	16.4	0.9	1.6×10^{-49}	5.2×10^{-32}	7.2×10^{-9}

ing changes to this behavior (i.e., disturbances) to be detected sooner.[2] Further research is required test whether the improved fault detection is still observed (1) with other types of process variability (e.g., pure stochastic variability without autocorrelation), (2) with other monitoring approaches (e.g., variable-wise unfolding instead of batch-wise unfolding), (3) with higher levels of sensor noise, and (4) in other processes. In addition, it should be investigated whether biological variability is tak If so, these observations could possibly be exploited to improve data-driven fault detection.

5. Conclusion

This paper presents RAYMOND, a new simulation package for generating RAYpresentative MONitorng Data available from `http://cit.kuleuven.be/biotec/raymond`.

RAYMOND is a flexible simulation package. Its design enables

- easy implementation of new processes to accommodate a wide range of applications,
- addition of process variability to the process model (e.g., biological variability or changing catalyst efficiency),
- inclusion of input fluctuations to represent upstream variations and non-perfect equipment,
- free specification of sensors, including measurement noise, sensor bias, and resolution,
- full control over simulated process faults, and
- easy switching between different control methodologies to test their respective performance.

RAYMOND can be used to test APMC techniques on a wide range of processes. In an extended case study, it was demonstrated that process variability has a significant influence on SPM, and should be included in case studies to obtain realistic and correct performance assessments, and fair comparison of various APMC techniques for monitoring, optimization, or control. The full control over process faults in RAYMOND enables testing of fault detection and rejection on a wide range of different process disturbances.

Acknowledgements

Work supported in part by Projects OT/10/035 and PFV/10/002 (OPTEC Optimization in Engineering Center) of the Research Council of the KU Leuven, Project KP/09/005 (SCORES4CHEM) of the Industrial Research Council of the KU Leuven, and the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian Federal Science Policy Office. J. Vanlaer and P. Van den Kerkhof are funded by a Ph.D grant of the Agency for Innovation by Science and Technology (IWT). J. Van Impe holds the chair Safety Engineering sponsored by the Belgian chemistry and life sciences

[2] Somewhat similar to how small left-right changes to a steering wheel could be used to detect a broken steering column in a car traveling along a straight road when the car doesn't respond with small left-right direction changes anymore. Otherwise, defective steering would only be noticed when a large turn must be made.

Table 4: Fault detection results for linearly increasing growth rate: mean and standard deviation of the detection delay, and p-values of the standardized t-tests that compare the mean detection time to the case without biological variability (p_0), with variability with $\delta_{bv} = 1-2\%$ (p_{1-2}), and with variability with $\delta_{bv} = 5-10\%$ (p_{5-10}).

σ_{bv}	Detection [h]		p-value		
	μ	σ	p_0	$p_{0.001}$	$p_{0.005}$
Gradual feed rate decrease					
0	28.9	4.8	–	–	–
1-2%	17.1	5.7	3.8×10^{-19}	–	–
5-10%	16.2	5.0	5.7×10^{-23}	3.9×10^{-1}	–
10-20%	16.5	5.0	2.6×10^{-22}	5.4×10^{-1}	7.9×10^{-1}
Gradual DO sensor failure					
0	20.8	0.9	–	–	–
1-2%	13.7	1.8	1.6×10^{-43}	–	–
5-10%	13.4	1.5	1.1×10^{-50}	3.0×10^{-1}	–
10-20%	13.9	1.1	8.1×10^{-56}	6.5×10^{-1}	6.9×10^{-2}

federation essenscia. The scientific responsibility is assumed by the authors.

References

[1] G. Birol, C. Ündey, A. Çinar, A modular simulation package for fed-batch fermentation: penicillin production, Comput Chem Eng 26 (2002) 1553–1565.
[2] J. Downs, E. Vogel, A plant-wide industrial process control problem, Comput Chem Eng 17(3) (1993) 245–255.
[3] G. Birol, A. Zamamiri, M. Hjortsø, Frequency analysis of autonomously oscillating yeast cultures, Process Biochem 35 (2000) 1085–1091.
[4] A. Zamamiri, Y. Zhang, M. Henson, M. Hjortsø, Dynamics analysis of an age distribution model of oscillating yeast cultures, Chem Eng Sci 57 (2002) 2169–2181.
[5] J. Lee, C. Yoo, I. Lee, Fault detection of batch processes using multiway kernel principal component analysis, Comput Chem Eng 28 (2004) 1837–1847.
[6] C. Yoo, J. Lee, P. Vanrolleghem, I. Lee, On-line monitoring of batch processes using multiway independent component analysis, Chemometr Intell Lab 71 (2004) 151–163.
[7] J. Chen, H. Chen, On-line batch process monitoring using MHMT-based MPCA, Chem Eng Sci 61 (2006) 3223–3239.
[8] C. Zhao, F. Wang, N. Lu, M. Jia, Stage-based soft-transition multiple PCA modeling and on-line monitoring strategy for batch processes, J Process Contr 17 (2007) 728–741.
[9] J. Yu, S. Qin, Multiway Gaussian mixture model based multiphase batch process monitoring, Ind Eng Chem Res 48 (2009) 8585–8594.
[10] M. Jia, F. Chu, F. Wang, W. Wang, On-line batch process monitoring using batch dynamic kernel principal component analysis, Chemometr Intell Lab 101 (2010) 110–122.
[11] I. Monroy, K. Villez, M. Graells, V. Venkatasubramanian, Fault diagnosis of a benchmark fermentation process: a comparative study of feature extraction and classification techniques, Bioprocess Biosyst Eng 35 (2012) 689–704.
[12] G. Gins, J. Vanlaer, J. Van Impe, Discriminating between critical and non-critical disturbances in (bio-)chemical batch processes using multi-model fault detection and end-quality prediction, Ind Eng Chem Res 51(38) (2012) 12375–12385.
[13] J. Wan, O. Marjanovic, B. Lennox, Disturbance rejection for the control of batch end-product quality using latent variable models, J Process Contr 22 (2012) 643–652.
[14] J. Vanlaer, G. Gins, J. Van Impe, Quality assessment of a variance estimator for Partial Least Squares prediction of batch-end quality, Comput Chem Eng 52 (2013) 230–239.
[15] L. Chiang, M. Kotanchek, A. Kordon, Fault diagnosis based on Fisher discriminant analysis and support vector machines, Comput Chem Eng 28 (2004) 1389–1401.
[16] E. Russel, L. Chiang, R. Braatz, Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis, Chemometr Intell Lab 51 (2000) 81–93.
[17] N. Ricker, Decentralized control of the Tennessee Eastman challenge process, J Process Contr 6(4) (1996) 205–221.
[18] N. Ricker, J. Lee, Nonlinear model predictive control of the Tennessee Eastman challenge process, Comput Chem Eng 19(9) (1995) 961–981.
[19] P. Lyman, C. Georgakis, Plant-wide control of the Tennessee Eastman problem, Comput Chem Eng 19(3) (1995) 321–331.
[20] W. Ku, R. Storer, C. Georgakis, Disturbance detection and isolation by dynamic principal component analysis, Chemometr Intell Lab 30 (1995) 179–196.
[21] M. Nihtila, J. Virkunnen, Practical identifiability of growth and substrate consumption models, Biotechnol Bioeng 19 (1977) 1831–1850.
[22] K. Klatt, S. Engell, Ruehrkesselreaktor mit parallel- und folgereaktion, in: S. Engell (Ed.), ONichtlineare Regelung - Methoden, Werkzeuge, Anwendungen. VDI-Berichte, Vol. 26, 1993, pp. 101–108.
[23] M. Henson, D. Seborg, Nonlinear Process Control, Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.
[24] M. Diehl, I. Uslu, R. Findeisen, S. Schwarzkopf, F. Allgwer, H. Bock, T. Brner, E. Gilles, A. Kienle, J. Schlder, E. Stein, Real-time optimization for large scale nonlinear processes: Nonlinear model predictive control of a high purity distillation

column, in: Groetschel, Krumke, Rambau (Eds.), Online Optimization of Large Scale Systems: State of the Art, 2001, pp. 363–383.

[25] J. Hahn, T. Edgar, An improved method for nonlinear model reduction using balancing of empirical gramians, Comput Chem Eng 26 (2002) 1379–1397.

[26] C. Ündey, S. Ertunç, A. Çinar, Online batch/fed-batch process performance monitoring, quality prediction, and variable-contribution analysis for diagnosis, Ind Eng Chem Res 42 (2003) 4645–4658.

[27] P. Geladi, B. Kowalski, Partial least-squares regression: a tutorial, Anal Chim Acta 185 (1986) 1–17.

[28] P. Nomikos, J. MacGregor, Monitoring batch processes using multiway principal component analysis, AIChE J 40(8) (1994) 1361–1375.

[29] F. Arteaga, A. Ferrer, Dealing with missing data in MSPC: several methods, different interpretations, some examples., J Chemometr 16 (2002) 8–10.

[30] B. Li, J. Morris, E. Martin, Model selection for partial least squares regression, Chemometr Intell Lab 64 (2002) 79–89.

[31] I. Jolliffe, Principal component analysis, Springer Verlag, New York, 1986.

[32] P. Nomikos, J. MacGregor, Multivariate SPC charts for monitoring batch processes, Technometrics 37(1) (1995) 41–59.

[33] P. Nomikos, J. MacGregor, Multiway partial least squares in monitoring batch processes, Chemometr Intell Lab 30 (1995) 97–108.

[34] S. Qin, Survey on data-driven industrial process monitoring and diagnosis, Annu Rev Contr 26 (2012) 220–234.

[35] Z. Ge, Z. Song, F. Gao, Review of recent research on data-based process monitoring, Ind Eng Chem Res 52 (2013) 3543–3562.